NASnews
July – August 1997 • Volume 2, Number 25

In This Issue...

Initial SGI Origin2000 Tests Show Promise for CFD Codes

Several projects are under way to fully explore the performance characteristics of the new Silicon Graphics, Inc. (SGI) Origin2000 system, installed at the NAS Facility last April. One of these efforts centered around porting and optimizing ARC3D, a well-known computational fluid dynamics (CFD) code. Preliminary results show promise for achieving very high sustained performance levels, which could bode well for several other production CFD codes that run on NAS systems.

To The Article...

Features of *Currents* have been incorporated into this newly redesigned version of *NAS News*. While *NAS News* is published bimonthly, the online version may include additional up-to-date news, and will emphasize scientific visualization, graphics, animation, audio, and video. You'll also find links to events and other information about the Numerical Aerospace Systems (NAS) Facility.

NASnews

▶ Main Menu July – August 1997 • Volume 2, Number 25

Initial SGI Origin2000 Tests Show Promise for CFD Codes

by James Taft

Several projects are under way to fully explore the performance characteristics of the new Silicon Graphics, Inc. (SGI) Origin2000 system, installed at the NAS Facility last April. One of these efforts centered around porting and optimizing ARC3D, a well-known computational fluid dynamics (CFD) code. Preliminary results show promise for achieving very high sustained performance levels, which could bode well for several other production CFD codes that run on NAS systems.

The system (purchased jointly by NASA's Data Assimilation Office and the Information Technology Program at Ames Research Center) is composed of 64 RISC-based central processing units (CPUs), with 16 gigabytes (GB) of globally addressable main memory, and 330 GB of available disk space. The system is one of the first to incorporate the Non-Uniform Memory Access (NUMA) architecture, a design that allows the construction of large systems with hundreds or even thousands of processors at a modest cost.

A major factor in purchasing the Origin2000 was its initial performance on ARC3D, which uses algorithms that, although popular, are known to be "unfriendly" to cache-based RISC systems. The version of ARC3D that was tested was a simple three-dimensional (3D) transient EULER variant using a single rectilinear grid. Differencing is done implicitly using an Alternating Direction Implicit (ADI) solver, which sweeps through each of the cardinal directions one at a time, with partial updating of the fields after each sweep. Historically, this solver has been very inefficient on cache-based systems.

Challenges in Optimizing CFD Codes

CFD codes are one of the toughest classes of problems to port to RISC-

[Web Help for Parallel Systems Users]

[Cray User Group Highlights]

based architectures. This is due to the relatively high ratio of data fetches to floating point operations that are typical of numerical algorithms. Excessive data fetches severely stress the cache and memory-management hardware. The problem is further compounded for NUMA systems due to variance in memory access times for remote fetches. In some cases, the situation becomes untenable.

Much experience with RISC systems and limited experience with NUMA systems shows that worthwhile performance can only be had by substantially rewriting core routines. Luckily, this isn't as onerous as it first sounds. In most cases, the majority of the computational burden occurs in just a few routines, so recoding can often be done in weeks or months instead of years.

ARC3D Ported in a 'Matter of Minutes'

The effort required to port the ARC3D code from the C90 to the Origin2000 was trivial -- in fact, it was completed in a matter of minutes. The code uses no external libraries and is syntactically Fortran 77 compliant. The biggest concern -- preserving 64-bit precision in the arithmetic -- was accomplished by specifying the **-r8** and **-i8** compiler flags.

The effort to optimize the ARC3D code was more extensive. About 1000 lines of code, involving five major routines, were rewritten over a period of about two months.

In general, the strategy for optimizing ARC3D on the Origin2000 was quite similar to the process used on the CRAY C90, although the details differed. First, the code was optimized for a single processor, followed by parallel optimizations across multiple processors. Because each Origin2000 processor is less powerful than a CRAY C90 or T90 processor, parallel efficiency is much more important for achieving good performance overall.

Parallel efficiency can only be achieved when careful attention is paid to ensuring that data accesses are as local as possible. This means the data must reside in the local processor's cache or local main memory. Most of the ARC3D optimization work was focused on ensuring this characteristic in the code.

Vector-oriented Optimization

It should be noted that both Cray and Origin2000 systems are "vector oriented." By design, RISC architectures are pipelined machines that perform well on the regular organization of data found in vector code. If the data could be contained in cache at all times, then classic Cray-optimized vector code would run well on RISC systems. However, Cray code typically "breaks" cache as a result of very long vectors and large datasets. The end result is that loops and common block structures must be reorganized to improve cache reuse.

Much of the editing work on the ARC3D code involved reversing the order of the indices in the field arrays and "fattening" loops by combining multiple 3D loops into a single 3D loop. In addition, the number of vector temporaries in the ADI calculation was greatly reduced.

Optimization Was 'Straightforward'

The ARC3D code was parallelized in a straightforward manner: The code was compiled with the **-mp** option to enable parallelism in the compile phase. This was combined with approximately ten **c$doacross** compiler directives, which instruct the compiler to generate code that will execute the next Fortran "do" loop in parallel. These directives were placed at key spots within the code for the greatest parallel efficiency. This involved decomposing the 3D problem into groups of 2D planes, with each group assigned to a dedicated processor.

The efficiency of the parallel decomposition was enhanced by the utilization of two Origin2000 data-distribution directives that allow users to improve the distribution of their data throughout the system. These features were invoked by setting two environment variables. The first specified the user's default page size for data. The second specificied the allocation policy for placing the data across the memories in the system.

The entire optimization effort consumed approximately two months, most of which was expended in learning the behavior of the system itself in response to different optimization approaches -- lessons that can be applied when optimizing other codes in the future.

Test Series Challenged Memory, Cache

Several different dataset sizes were considered during the test series.

Ultimately, the largest problem was selected for the majority of tests because it fully stressed the memory and cache systems and offered the worst case scenario. The grid for this problem was defined to be 194x194x194 (7,301,384 grid points). This size ensured that the problem would not fit into the aggregate of the cache memories and that the data would be scattered across many of the local memories in the NUMA-based Origin2000.

The test runs consisted of executing two timesteps on varying numbers of processors, and performing a residual error check to verify the accuracy of the solution. Performance timing spanned only the work performed for each timestep and the residual error calculation. A subsidiary test series of twenty timesteps verified that the timings gleaned from the two timestep runs were accurate and would linearly scale to higher timestep counts.

Preliminary Results 'Impressive'

Preliminary results on the 64-CPU Origin2000 are impressive. In a nutshell, ARC3D an historically "cache-abhorrent" CFD code was restructured with no change in the fundamental algorithms to yield over 6.3 gigaflops per second (Gflop/s) in sustained performance equivalent to past performance of this code on a 16-CPU C90.

Results for the 194x194x194 problem (1 and 64 CPUs) are presented in Figure 1, below. As can be seen, the major time consumers were the x-, y-, and zdirect routines, which continue to scale well on 64 CPUs. One of the strange artifacts of the NUMA environment in this particular case is that the zdirect routine scaled better than the xdirect routine, even though the xdirect routine is unit stride through the data and the zdirect routine is the worst stride through the data.

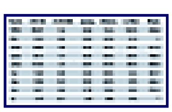


Figure 1

The overall ARC3D scaling for executions on 1 to 64 CPUs is presented in Figure 2, below. This chart shows virtually no scaling fall-off as the number of processors increases, and indicates that substantial gains are still to be had.

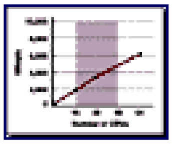


Figure 2

Continued Performance Improvements

Cursory examination of the ARC3D test results indicates that even better performance can be obtained simply by executing the code on systems with more processors. The current code utilizes about 100 of the total available Mflop/s per processor (peak is 380). Further optimization could improve this performance, resulting in even higher sustained Gflop/s ratings. The Origin2000 architecture may finally allow successful migration from traditional "vector" systems for some classes of production codes. Further tests on 128-CPU systems will better define the ultimate scalability.

For more information about porting and optimizing code for the Origin2000, send email to jtaft@nas.nasa.gov.

NASnews
▶ Main Menu
July – August 1997 • Volume 2, Number 25

This Graphic...

Initial Tests Show Promise

Return to Article...

In This Issue...

SGI Origin2000 Shows Promise

IPv6, New Internet Protocol

Large CFD Code Successfully Ported to Cray Fortran 90

Metacenter Success, Challenges

Parallel Supercomputers: Price vs. Complexity

Portable Code Optimization Study

Web Help for Parallel Systems Users

Cray User Group Highlights

Initial SGI Origin2000 Tests Show Promise for CFD Codes

Routine	1-CPU Wall	64-CPU Wall	Speedup	Efficiency	% of Work	Mflop/s
TOTAL	364.47	7.45	48.93	76.5%	100.0%	6309.8
step	13.83	0.60	23.05	36.0%	8.1%	—
xdirect	83.66	1.66	50.39	78.7%	22.3%	6914.5
ydirect	88.14	1.51	58.20	90.9%	20.3%	7580.4
zdirect	110.06	1.97	55.87	87.3%	26.4%	5828.3
rhsx	15.92	0.40	40.20	62.8%	5.3%	4813.7
rhsy	18.86	0.44	42.62	66.6%	5.9%	4308.1
rhsz	33.27	0.69	47.88	74.8%	9.3%	2743.4
bc	0.75	0.17	4.40	6.9%	2.3%	—

Summary of the ARC3D code performance for 1 and 64 CPUs on the Silicon Graphics Inc. Origin2000 system at the NAS Facility. The table shows elapsed wall-clock time and Mflop/s (millions of floating-point operations per second) in performance for the entire run (in the "Total" row), as well as each of the major routines contributing to the run time.

Return to main article

NASnews

▶ Main Menu July – August 1997 • Volume 2, Number 25

Large CFD Code Successfully Ported to Cray Fortran 90

by Peter Gnoffo and Art Lazanoff

Peter Gnoffo and Art Lazanoff describe their success with porting the well-known LAURA code from CF77, Cray Research Inc.'s old Fortran compiler, to the latest version, F90. LAURA, the Langley Aerothermodynamic Upwind Relaxation Algorithm, is a program for simulating thermochemical nonequilibrium flows over vehicles in hypersonic flight. Because Cray stopped supporting CF77 over a year ago, the authors took on the task of solving one LAURA researcher's problem with CF77 by testing the waters with this complex working code.

Recently, a researcher involved with the LAURA project at NASA Langley Research Center (LaRC), encountered a problem with running the old Cray Fortran compiling language (CF77) on the NAS Facility's CRAY C90 supercomputer. Compiling a 175-block case caused a symbol table overflow in CF77.

While working to identify the nature of the compiler problem and consider alternatives, a workaround was devised to solve the immediate problem for the LAURA researcher. At the same time, another researcher using the LAURA code had an even larger problem, which needed yet a different workaround. It was apparent that a real solution to these problems -- which was sure to appear again -- was needed.

The consensus among the NAS Facility's scientific consultants and high-speed processor group was to investigate porting the LAURA code to the new Cray Fortran 90 compiler (F90), which manages symbol tables differently. This would either solve the problem, or at the very least, produce a bug that could be documented in a supported product.

Porting Risks and Concerns

LAURA uses a preprocessor to build source code that is compiled based on user inputs about the problem being analyzed (for example, block size, species participants, or thermal state of gas). In addition, LAURA is set up to run on a variety of platforms. Porting a complex code such as this from one version of a language to another -- especially when going from an older, stable compiler to a newer one -- entails a number of risks, including code corruption, incorrect results, and poor performance.

A major concern was whether F90 supported macrotasking -- a critical feature in the LAURA code, one of the few in CFD that takes advantage of macrotasking options. It was quickly determined that F90 supports the identical syntax and functionality of CF77. This was verified by running a matrix multiply test.

Few Changes Needed

Before we tackled the problem with CF77's table symbols, a copy of LAURA 4.4 was used in an initial round of porting and testing, using basic examples described in the LAURA user documentation. The preprocessor and main logic were compiled and run with changes only to the **make** files, in order to invoke F90 rather than CF77. The results were correct and timings were comparable. However, the test cases were too small to do meaningful timing comparisons; additionally, the test cases did not perform any multitasking.

A much larger LAURA code was obtained and also passed correctness and functionality tests. Only one line of code had to be changed for correct macrotasking execution; this involved the initialization of the IPCU variable, used to keep track of the CPU that a task is running in. The original DATA statement made ICPU a globally available variable, but ICPU needed to remain a private variable within a task. Fixing this was a matter of converting the DATA statement from:

```
DATA ICPU/0/
```

to

```
ICPU = 0
```

At this point, LAURA. 4.4 worked correctly with the examples tested. The next step was to try out the case that overran CF77's symbol tables. F90 successfully compiled the code "as is." The problem was so large that F90 used just over 40 megawords (MW) -- 2 to 3 MW is typical -- for the compilation, with optimization turned on.

Two tests were used to execute the problem. The first was an in-memory test that required 230 MW for execution. The second test used solid-state disc memory to hold solution Jacobians (implicit matrices) for each grid block. This approach reduced the memory requirement to less than 120 MW.

The initial workaround for the symbol-table limitation of CF77 took more than a week to produce. F90 compiled the 175-block case as is and ran the code somewhat faster than the CF77 compiler could run the workaround. The workaround and the "fix" were completed at the same time, with just a few changes. Significantly, the code changes were minor, and these particular changes did not affect LAURA's ability to run on other platforms.

Benefits of New Features, Support

F90 supports several new features, including standardized dynamic array allocation. We are now investigating ways to take advantage of these features to enhance LAURA's capability and maintainability. And now that the LAURA code has been successfully compiled on F90, the NAS scientific consultants can enlist Cray's help in this effort.

For details on using F90, see the online tutorial created by the scientific consulting group. More information on the LAURA project is also available.

NASnews

▶ Main Menu July – August 1997 • Volume 2, Number 25

Langley-Ames SP2 Metacenter Enjoys Success, Plans for New Challenges

by James P. Jones and Mary Hultquist

Two members of the Langley-Ames metacenter team report on progress and plans for the future. The project, which connects two IBM SP2 testbed systems at NASA Langley Research Center (LaRC) and the NAS Facility at Ames Research Center, is funded by NASA's High Performance Computing and Communications Program.

The Langley-Ames metacenter team has achieved numerous goals since the project's inception in late 1995: getting faster job turnaround; balancing the workload across both systems; providing a wider range of resources; and migrating jobs automatically, giving users the ability to direct or limit migration.

These successes have attracted the attention of other sites, including Wright-Patterson Air Force Base and the U.S. Army Corps of Engineers, Vicksburg, MS. Team members at LaRC and Ames have begun transferring critical software used in the metacenter to these sites, including the Portable Batch System (PBS, the metacenter's batch queuing system), NAS Site Wide Accounting (ACCT++), and the PBS external job schedulers written at NAS and LaRC.

Focus on Access, Software

Since October, when the metacenter officially became the primary means for users to access the IBM SP2s, software development efforts have focused on two major areas: improving access to user files and data (which may reside on either system) and identifying critical software to be transferred to the next version of the metacenter testbed.

Fast, Reliable Access Not So Easy

The largest technical challenge has been to provide users with fast, reliable access to files located across the metacenter. Currently, users must track their file locations, as those files can't be seen from other systems. The best solution would be to have all users' SP2 files visible and accessible from anywhere within the metacenter. However, given the immaturity of software on the testbed systems, providing this capability is easier said than done.

The two most promising distributed filesystems, NFSv3 and DCE/DFS, are not yet available and not mature enough, respectively, to reliably fill this need. Metacenter team members are working with IBM and other vendors to provide this global accessibility. As an interim solution, the team is using PBS, which provides a file-staging capability (a mechanism for users to identify files that are needed by their batch jobs). PBS handles the copying of files to the appropriate system within the metacenter, and returns solutions to the location specified by the user.

In addition, the team has modified PBS, the job scheduler, and the network configuration within and between the two SP2s to improve file transfers. While this approach has worked, it is not the most graceful solution, because it requires that users specify all files to be used during a job. The team is pursuing the elusive goal of a global shared filesystem as it works with vendors and develops local software.

Sharing Experiences, Planning for Future

While continuing work on these enhancements, the metacenter team is sharing its experiences with other interested sites through invited talks and visits. The team is also planning the transition to the next HPCC-funded testbed, which will replace the current SP2s with yet-to-be-determined systems. This "new" metacenter will expand to a include a third parallel system located at NASA Lewis Research Center. All three systems are scheduled to arrive by the end of the year.

Most options for the next system include architectures with a single system image -- that is, each supercomputer appears to users as one system, with contiguous processors and memory, rather than as a cluster of machines, each with their own CPUs and memory.

A shift in the underlying architecture alone could alleviate much of the difficulty experienced with the current metacenter. In addition, solutions for locating and transferring data files for processing will be designed into the new configuration. (The SP2s were originally configured independently before becoming part of the metacenter.) The next testbed will be specifically configured with the metacenter in mind.

- More information on the metacenter project.

NASnews

▶ Main Menu July – August 1997 • Volume 2, Number 25

Cache-based Parallel Supercomputers Trade Price for Complexity

by Ayse Sercan

Most numerically intensive application codes being used today have been optimized for a single processor of a vector supercomputer. While vector machines, with their combination of proprietary processors and high-speed, high-capacity memory have proven very successful, it is expected that in the future the majority of supercomputers will be parallel systems that are based on large numbers of commodity, cache-based processors coupled with slower, commodity memory systems.

This change in architecture will have an impact on the way supercomputing applications are written, not only because the software must be able to use more than one processor at a time in order to replicate the processing power of a traditional vector machine, but because the processor and memory architectures of cache-based and vector-based systems are so different.

On the surface, it would appear straightforward to produce code that runs efficiently on any cache-based system. Although some of the important computational algorithms used at the NAS Facility require different programming styles on vector- and cache-based machines, neither architecture class appears to be favored by particular algorithms, in principle. But practice shows that the situation is more complicated.

Issues With CFD Applications

Many of the application areas of interest to researchers who use the NAS Facility, such as computational fluid dynamics (CFD), deal with algorithms and computer codes that are "memory bound" rather than "processor bound"; that is, they require large, fast memory more than processor power. The trick in optimizing these codes is to ensure that data is constantly being fed to the processor from memory, and that the processor is never "stalling" or "idling" while it waits for data. On traditional vector systems, with fast memories and high-bandwidth interconnects, this has been easy to do. But the same is not true for the new breed of parallel supercomputers.

Two steps are required for optimizing a program to run efficiently on these machines. The first is to tune the program for good single-processor performance. After this has been achieved, one can then address the issues related to harnessing multiple processors to work together efficiently within the program.

At the NAS Facility, a study was conducted to determine exactly what type of single-processor code-tuning would be required to achieve a code that was efficient and portable across a variety of current microprocessors. All the microprocessors studied are used as the building blocks in various parallel supercomputers.

The results, published in technical report NAS-97012 in April, includes several counterintuitive results that serve as a cautionary reminder that parallel programming and memory accesses are not the only factors that determine performance, and that within the class of cache-based systems, significant differences exist. (For a more technical look at the code optimizations tested in this study, see "Study Shows Poor Results for Portable Code Optimizations on Cache-based Processors".)

Making Use of Commodity Components

One of the benefits of a distributed or parallel system -- one in which there are several smaller computers sharing the work, rather than one large computer -- is its scalability and cost.

Several computer vendors now build distributed supercomputers based on multiple cache-based microprocessors; specifically, reduced-instruction set computer (RISC) processors like those used in IBM PowerPCs and Unix workstations. These distributed systems can, in principle, be scaled up easily by adding more processing power as needed.

Because the components are commodity items that are readily and cheaply available, such systems are also priced attractively. The challenge to users is that they must design their software to maximize the use of multiple processors and to handle changes in the number of processors available.

In addition, most microprocessors today are superscalar; they can issue several arithmetic instructions per clock cycle if there are enough independent instructions available. To take advantage of this, programmers again need to optimize the movement of data into the processor.

Cheap, Slow Memory and Smaller Cache

Vector-based computers have expensive, fast memory and specialized vector registers. In order to take advantage of this, programmers must use inner-loop independence and send arrays to the processor in regular increments or strides. On the other hand, RISC processors are cache-based systems, with cheap but relatively slow main memory and small, expensive high-speed memory buffers (caches). To take advantage of these systems, programmers must exploit the spatial and temporal locality of the data.

Data locality is key to ensuring that blocks of data, once fetched from main memory and stored in the cache, are used as much as possible before being written back to main memory or being removed to make room for new data that the processor needs.

Programming for Cache-based Systems

In order to take advantage of cache-based systems, programmers should follow a few basic guidelines, many of which are appropriate for vector computers also. These include the use of "cache-friendly" array operations, such as those featuring unit stride or zero stride, and programming loops to perform as many operations as possible on the cached data. If unit strides are not possible, then programmers should be careful to avoid "bad" strides such as large powers of 2, just as they would on vector computers.

The tests conducted at the NAS Facility have shown that optimizing code on commercial, cache-based systems is not simple and straightforward, as noted in the article below. Because highly detailed knowledge of the system is needed in order to optimize code, and because microprocessor architectures are so different, chances are slim for making that code portable to other parallel systems.

NASnews

▶ Main Menu July – August 1997 • Volume 2, Number 25

Study Shows Poor Results for Portable Code Optimizations on Cache-based Processors

by Rob F. Van der Wijngaart

To test the truth of the assumptions about creating good, portable code that is optimized for cache-based systems, members of the NAS Facility's algorithms, architectures, and applications group used part of the NAS Parallel Benchmarks 2 (NPB 2) program suite, to benchmark performance on four cache-based processors used in parallel supercomputers. (See "Cache-based Parallel Supercomputers Trade Price for Complexity").

The Scalar Penta-diagonal (SP) code in the NPB 2 suite provides a stress test on the memory system of computers, because there are relatively few operations per grid point in any of its loops. The most critical segment is the line solver, which solves independent systems of linear equations associated with grid lines. Because there are three families of grid lines in three-dimensional space, there are also three different solver routines, one for each of the three "factors."

These three routines, with the generic names x-loop, y-loop, and z-loop, were used as examples of codes to be run on cache-based systems. Each routine was executed for four grid sizes: 16^3, 32^3, 64^3, and 80^3 points. To avoid potential problems with power of two ("bad") strides, all array grid dimensions were padded by 1.

Code Example Shows Progression

The baseline code shown in the example is identical to the one currently used in NPB 2. Four successive, cumulative optimizations were applied to these routines, as follows:

The first optimization eliminated temporary variables for incremented

[Price vs. Complexity]

Portable Code Optimization Study

[Web Help for Parallel Systems Users]

[Cray User Group Highlights]

indices; that is:

```
i+1, i+2, j+1, j+2, k+1, k+2
```

instead of

```
i1, i2, j1, j2, k1, k2
```

This allows good compilers to better allocate the memory registers.

Second, in order to reduce loop overhead, inner loops of fixed length (such as m=1,2,3) were unrolled (written out by hand).

In the third optimization, the fourth index (m) of the arrays **rhs** and **lhs** was moved to the first position. This improves spatial data locality, because each iteration of the inner loop uses all three elements of **rhs** and all five elements of **lhs** at each grid point.

Finally, to increase the number of independent computations in the inner loop, the first available loop (j for x-loop, i for both y-loop and z-loop) was unrolled to a level of two.

Alternative Optimizations of Kernel Code

In the cases of y- and z-loop, there were two plausible code variations, both pertaining to loop order. The previous optimizations all used "canonical" loop orderings; namely, running index k, j, and i for the outer, middle, and inner loop, respectively. However, computationally, the most "natural" ordering would be to process a whole grid line in the innermost loop before moving to the next grid line.

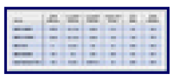
The first alternative optimization strategy was to use j and k as the running index for y- and z-loop, respectively, while keeping the unrollings of the first available loop. This causes large strides in the arrays, which generally is detrimental to performance. The benefit is that there is some overlap in subsequent iterations, because there is a recursion in the grid line direction in the solver algorithm. The running index for the middle loop in each of the two loop nests is always I.

The second alternative used the same natural loop order as in the previous optimization, but the unrolling optimization was eliminated.

Variety of Architecures Examined

Detailed results of the effect of these optimizations on performance for a variety of microprocessor architectures, in Mflop/s (millions of floating point operations per second), are available. Some interesting observations can be made, based on the test results.

The microprocessors used in the study were for the most part RISC (reduced instruction set computer) based: a Silicon Graphics Inc. (SGI) MIPS R8000 and R10000; a Digital Equipment Corp. (DEC) Alpha EV5; and an IBM Corp. RS6000. The one system based on the complex instruction set computing (CISC) architecture was a 200-megahertz (MHz) Intel Corp. Pentium Pro (see table below for details.)



All of these RISC systems are currently used in parallel platforms; the MIPS R8000 and R10000, the IBM RS6000, and the Intel Pentium Pro are all used at the NAS Facility, in the SGI PowerChallenge and Origin clusters, the IBM SP2, and the massively parallel Whitney system, respectively. The Alpha EV5 is used in the CRAY T3E.

SGI MIPS R8000

On the MIPS R8000, unrolling the i-loops for the y- and z-factors greatly deteriorated performance, hinting at a problem with the allocation of registers for the fattened loop body. Processor performance was reduced for large grids (sizes 64 and 80) that did not fit completely in cache. Apparently, the memory bandwidth was not sufficient to fill the cache fast enough from main memory. It was also interesting to note that the x-factor -- generally the most cache-friendly of the three factors -- performed worse than the y- and z-factors. This peculiar phenomenon, which was not observed in other members of the MIPS processor family, may have been caused by the interleaved structure of the MIPS R8000's cache.

Intel Pentium Pro

The 200-MHz Intel Pentium Pro processor displayed great sensitivity to problem size with respect to optimizations of the z-factor. But more importantly, with proper tuning it maintained a performance of about

20 MFLOP/s for a nontrivial floating-point computation. This places the Pentium Pro squarely in the scientific workstation performance range.

DEC Alpha EV5

For the DEC Alpha EV5, performance of the y-factor degraded significantly when switching to the natural loop order, whereas for the z-factor the effect was mixed. The largest performance improvement for all factors came from moving the m-index to the first array position. Unrolling the i- or j-loops, however, had little or no positive effect. The EV5 showed -- more or less as expected, as it featured the greatest data locality -- its best overall behavior for the x-factor with m as the first array index.

IBM RS6000

The IBM RS6000 -- somewhat surprisingly, as it was expected that the optimization would have the same impact on all three routines -- benefited from eliminating the auxiliary variables i1 and i2 in the x-factor, whereas a similar program change for the y- and z-factors had a *negative* effect. In addition, it was noticed that the best performance for the x-factor required a partially unrolled j-loop; optimal performance for the y- and z-factors was achieved without unrolling the i-, j-, and k-loops.

Finally, moving the m-index proved to be positive for the x-, neutral for the y-, and negative for the z-loop. It was concluded that there is a relatively small effect of grid size on performance, indicating that for this processor, memory bandwidth is sufficient to feed the cache.

Counter-intuitive Performance Results

Several of the performance results were unexpected. In particular, the beneficial effect of the natural loop ordering for the z-factor for several processors was surprising. Apparently, the NAS team's intuition about what constitutes good cache code was inadequate. In order to get a better understanding of what was happening in the cache, the cache usage of the different pieces of code on the four processors was simulated, using a model that incorporated only total cache size, line size, and associativity for each system.

The cache simulations showed that, generally, there was very poor correlation between measured performance and cache misses. Most strikingly, grid sizes 16^3 and 32^3 fit entirely in the cache of the MIPS R8000, so no misses occurred at all, but performances varied by a factor of three.

Less dramatic -- but equally vexing -- was the *negative* correlation between measured performance and calculated cache misses for the y-factors on the Pentium Pro, DEC Alpha, and IBM RS6000. It was noted, however, that on several processors the unexpected beneficial effect of using the natural loop ordering for the z-factor was borne out by the cache simulations.

Disappointing Conclusions

Evidently, there is no single optimization strategy that gives the best performance for all grid sizes for all processors. Even if attention is restricted to one processor at a time, variable system architectures still do not allow a single uniform optimization strategy. It was also observed that the scatter in the optimal tuning strategies is substantially larger for the y- and z-factors than for the x-factor.

The counter-intuitive results led to the conclusion that the programmer's work in tuning code for higher performance in cache-based, commercial processors requires so much knowledge and information about the entire configuration (user program, problem parameters, system software, and hardware organization), that it is practically impossible to produce good code that is also portable.

In other words, it is impossible to tell whether a piece of numerical software will perform well simply by examining the code itself and the system on which it will be run. General optimization strategies designed to take advantage of cache do not by themselves yield high performance. Other factors, such as software pipelining, need to be considered as well when designing efficient codes, but this will hamper portability of codes between architectures.

NASnews

Main Menu

July – August 1997 • Volume 2, Number 25

This Graphic...

Poor Results for Code Optimizations

Return to Article...

In This Issue...

Study Shows Poor Results for Portable Code Optimizations on Cache-based Processors

Name	RAM MBytes	L1-cache KBytes	L2-cache KBytes	cache line Bytes	CPU MHz	Peak Mflop/s
MIPS R8000	4096	16i+16d	4096	512	90	360
MIPS R10000	4096	32i+32d	4096	128	195	390
DEC EV5	8	8i+8d	96	32	300	600
IBM RS6000	128	32i	256	256	66	267
Intel Pentium Pro	128	8i+8d	256/512	32	200	200

One of the complications presented by cached-based systems is a wide variety of configurations. A study conducted last spring at the NAS Facility included tests to determine the efficacy of creating efficient, portable code for these systems. The systems used in the tests, as shown above, ranged from 8 megabytes to 4 gigabytes of RAM, and from a processor speed of 300 megahertz to a mere 66 megahertz. This variety of configurations makes creating portable, optimized code not just difficult, but almost impossible, the study concluded.

Return to main article

Help for Parallel Systems Users Available on the Web

by Cristy Brickell

The World Wide Web is an often overlooked source of information for researchers who use the NAS Facility's parallel systems. The NAS web site includes a wide variety of information, including mailing list archives and pages for reporting problems and workarounds.

Among the most useful pages for users are those that explain how to use PBS, the Portable Batch System. Included are instructions for setting up environments for PBS, examples of submitting and running batch and interactive jobs, and commands for checking the status of PBS jobs. The accounting process and scheduling policies for each system are also provided.

Additional PBS documentation, with online tutorials in an easy-to-use format, is available from the NAS Facility's scientific consulting web pages.

Tutorial topics include: using PBS; converting batch jobs from Cray's NQS to PBS; and using XPBS (the graphical user interface PBS tool).

Mailing lists for both the IBM SP2 (babbage) and the Silicon Graphics Inc. (SGI) Power Challenge cluster (davinci) allow the parallel systems group to keep users informed about system status and changes. Note that the information in these lists is only available to users of that system.

- Systems status information
- Message of the Day for each system
- Schedule of planned maintenance for each system

The Problems and Workarounds web pages allow users to communicate with each other and with the NAS Facility staff about running jobs on the parallel systems. For example, several davinci users noticed that the **fsplit** command was occasionally truncating their source files. Using a form available on each system's corresponding Problems and Workarounds page, they submitted examples to the system administrator, who then placed instructions for avoiding this problem on davinci's workaround page.

Users can also submit their own workarounds to problems found on the following systems: the Langley-Ames metacenter (two IBM SP2s, poseidon and babbage, respectively), davinci, and the SGI Origin2000 (turing).

Personal assistance from the NAS User Services staff continues to be available 24 hours a day, 7 days a week. Call (415) 604-4444 or (1 800) 331-8737, or send email to support@nas.nasa.gov.

Cray User Group Highlights: Future, Fire, and Flight Tests

by Ayse Sercan

The 39th meeting of the Cray User Group was a resounding success, according to both first-time and veteran CUG goers. The conference, co-hosted by the NAS Systems Division and Sterling Software Inc., was held May 5-9 in San Jose.

Highlights included eye-opening presentations from keynote speaker Richard Hart and Ames researcher Robert L. Meakin. In addition, Ed McCracken, president and CEO of Silicon Graphics Inc. (SGI), shared insights into the "new" SGI-Cray Research partnership.

The Future of Supercomputing

On May 5, Jack Hansen, associate director of information technology at Ames Research Center, welcomed participants. Hansen noted that Ames is "lucky to have the tremendous resources of Silicon Valley at our doorstep," and outlined four areas where supercomputing will dominate in the future:

- Simulations of manufacturing processes will let manufacturers "speed through prototyping without building expensive and time-consuming models, and will give them more data about the process than previous methods."
- Improved aircraft/spacecraft operations will ease the strain of increasing air traffic.
- Scientists will use supercomputers for studying Earth (with EOS, the Earth-Observing System) and space (with satellites and probes).

  "EOS will give earth scientists incredible amounts of data about the planet," while supercomputers will give space scientists interpretive capabilities to help them "better understand the limited data that comes back on the low-bandwidth transmissions."

- Improvements in system autonomy and fault tolerance will give computers "alternatives to failure when something goes wrong."

Media Role in Scientific Collaboration

The keynote address, delivered by Richard Hart, executive producer at c/net TV and c|net Online, focused on collaboration in scientific venture and on how to make people passionate about science.

"Scientific research *is* interesting to mainstream audiences," Hart maintained. "You don't need to understand all the details about a technology to find it interesting. What people find interesting and entertaining is a story." To prove his point, he had one unsuspecting audience member hold a highly insulating materialwhich Hart promtly blowtorched. Both the material and the surprised "volunteer" came away unscathed.

Hart suggested that scientists discuss their work not only in scientific journals, but also with the mainstream press. "Everyone is interested in technology and sharing information about your work widens the circle of collaboration," he said.

"Never be afraid to contact the mainstream press about your research," Hart said. "De-specializing information delivery allows a wider audience to see what you are doing and invites responses."

As an example, Hart cited one researcher who created a material that hardens instantly when struck, but is soft and flexible until that point. Although the researcher had no practical application for the material, a company that saw the product mentioned in a news report contacted himand the material is now being used to design bulletproof shirts.

Among Hart's suggestions for widening the circle of collaborators: Hold interdepartmental gatherings (formal or informal), which can lead to "innovative answers or interesting questions about research." He also suggested working in teams of interdisciplinary experts (particularly easy to do in Silicon Valley, Hart noted), which can result in "amazing discoveries."

"Get people excited about your work and wonderful things happen. The most exciting innovations coming out today didn't come from a marketing scheme cooked up in a conference room, but from the passion of the researcher," Hart concluded.

SGI's Commitment to Supercomputing

On May 6, Ed McCracken, president and CEO of Silicon Graphics Inc., spoke about the future for the recently combined SGI and Cray Research Inc. (the two organizations merged in July 1996.)

"We are very committed to the supercomputer field and are very happy about the impact of Cray Research on our overall company," he remarked.

In the months following the Cray merger, McCracken said, SGI took a "long hard look" at themselves and defined five corporate values:

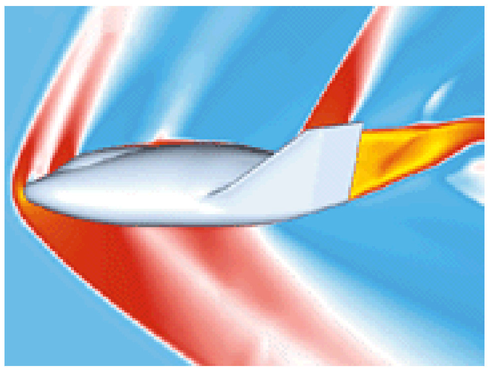
- Innovation, including, "both innovative products and encouraging innovation in our customers."
- Fairness and respect for employees and customers.
- Integrity, which means "meeting commitments," a value which McCracken attributes to Cray's influence.
- Passion for the technology and for the customers' work.
- Delivering breakthrough results, because, McCracken explained, "none of the above mean anything if there's no product."

Simulations Save Costly Test Flights

Robert L. Meakin, of the U.S. Army Aeroflightdynamics Directorate at Ames, discussed his work on simulations of unsteady, three-dimensional viscous flow.

Meakin illustrated the need for such simulations through dramatic footage of test flights, in which the bomb bay on a test plane opened, bombs dropped out, hit the air stream under the planeand then boomeranged back into the planesometimes knocking parts of the plane loose.

"It is important to be able to predict in advance the aerodynamics of an application," Meakin stated, because "flight testing is highly risky for the pilot and aircraft, and is also costly." Aerodynamics can be predicted through testing in wind tunnels or on supercomputers like those at the NAS Facility.



This simulation of the Mach field around a full-scale X-38 vehicle in low supersonic flight was part of a presentation by Robert L. Meakin at the 39th Cray Users Group (CUG) conference, San Jose, May 5-9. Meakin, a researcher at Ames Research Center, discussed the role of supercomputers in aerodynamics. The conditions of this simulation included a Mach 1.5 freestream, 15-degree angle of attack, and a Reynolds number of 25 million, and was generated using adaptive spatial partitioning and refinement for overset structured grids. According to Meakin, results of the adaptive solution "compare very favorably with a high-fidelity conventional Chimera result obtained independently by researchers at NASA Johnson Space Center." The adaptive scheme is also being applied to a variety of defense applications as part of the Department of Defense High Performance Computing Modernization Program.

According to Meakin, such simulation is best done on supercomputers because "the geometry is complicated, and the physics are complicated and unsteady." On a supercomputer, researchers can model such a system using the Chimera approach, which trades "one complicated geometric domain for several less-complicated parts that overlap," he said.

A benefit of the Chimera approach is that it allows arbitrary holes in grid systems, which provides an added degree of flexibility from a grid-generation standpoint. The cost of the advantages gained through this approach is reflected in the need to establish domain connectivity among overlapping grid components. It is important that this operation be done efficiently, Meakin explained, because domain connectivity must be established at every timestep for problems that involve relative motion between component parts.

Meakin also described the role of the NAS Facility supercomputers for simulations in his own research, which centers on the aerodynamics of helicopters for civil and military applications.

Information about this and other CUG conferences is available. The next meeting will be held next spring in Stuttgart, Germany.

Credits

Executive Editor: Marisa Chancellor

Editor: Jill Dunbar

Staff Writer: Ayse Sercan

Contributing Writers: Cristy Brickell, Peter Gnoffo, Mary Hultquist, James P. Jones, Art Lazanoff, James Taft, Rob F. Van der Wijngaart

Graphics Coordinator: Chris Gong

Online Page Layout and Graphics: Joel Antipuesto, Fay Pattee, Chris Gong, Rosemary Wadden

Other Contributors: Peter Adams, Richard Anderson, Bob Ciotti, James Donald, Andrew Hoag, Robert Hood, Kevin Lahey, Robert L. Meakin, Terry Nelson, Fay Pattee, Marcia Redmond, Subhash Saini, Deepak Srivastavana, Leigh Ann Tanner, Maurice Yarrow

Editorial Board: Cristy Brickell, Marisa Chancellor, Jill Dunbar, Kevin Lahey, Nateri Madavan, Patrick Moran, George Myers, Fay Pattee, Ayse Sercan, Harry Waddell

Special thanks to former senior writer Elisabeth Wechsler, who contributed her energy and talent to NAS News for more than three years.

July – August 1997 • Volume 2, Number 25

In This Issue...

Initial SGI Origin2000 Tests Show Promise for CFD Codes

Several projects are under way to fully explore the performance characteristics of the new Silicon Graphics, Inc. (SGI) Origin2000 system, installed at the NAS Facility last April. One of these efforts centered around porting and optimizing ARC3D, a well-known computational fluid dynamics (CFD) code. Preliminary results show promise for achieving very high sustained performance levels, which could bode well for several other production CFD codes that run on NAS systems.

To The Article...

Features of *Currents* have been incorporated into this newly redesigned version of *NAS News*. While *NAS News* is published bimonthly, the online version may include additional up-to-date news, and will emphasize scientific visualization, graphics, animation, audio, and video. You'll also find links to events and other information about the Numerical Aerospace Systems (NAS) Facility.